Developing Scalable Applications using Portable Extensible Toolkit for Scientific Computation (PETSc) for the Next-Generation Power Grid

Shrirang Abhyankar

Mathematics and Computer Science Division
Argonne National Laboratory

November 15, 2011

Outline

Outline

Next-Generation Power Grid

- Next-generation power grid
  - PMUs, smart meters, Distributed Generation, Plug-in hybrid vehicles, Smart and Micro-grids, Power electronics, Increased communication
- Computational challenges
  - Data explosion , Real-time simulation requirements , Larger or denser network , Multi-scale (temporal, geographical)

Can we develop applications for large systems that run in real-time and capture behavior at a sub-cycle level?

Next-Generation Power Grid

- Next-generation power grid
  - PMUs, smart meters, Distributed Generation, Plug-in hybrid vehicles, Smart and Micro-grids, Power electronics, Increased communication
- Computational challenges
  - Data explosion , Real-time simulation requirements , Larger or denser network , Multi-scale (temporal, geographical)

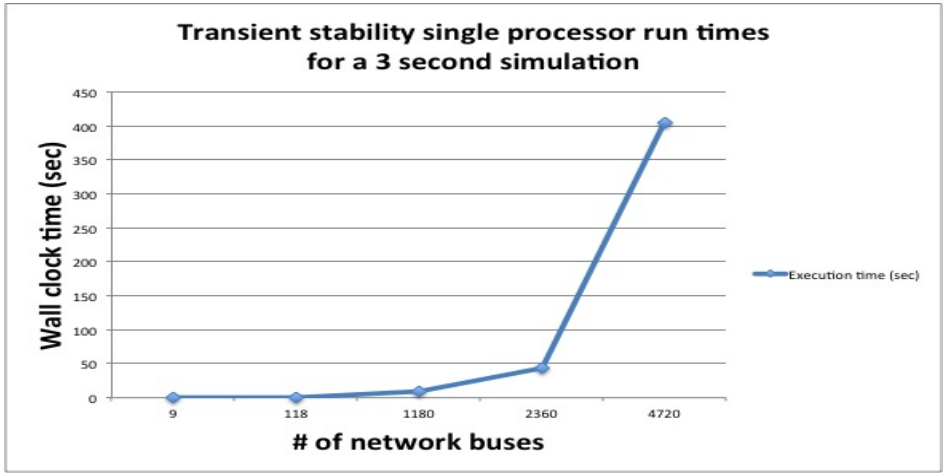
Can we develop applications for large systems that run in real-time and capture behavior at a sub-cycle level?

Next-Generation Power Grid

- Next-generation power grid
  - PMUs, smart meters, Distributed Generation, Plug-in hybrid vehicles, Smart and Micro-grids, Power electronics, Increased communication
- Computational challenges
  - Data explosion , Real-time simulation requirements , Larger or denser network , Multi-scale (temporal, geographical)

Can we develop applications for large systems that run in real-time and capture behavior at a sub-cycle level?

Computational challenges



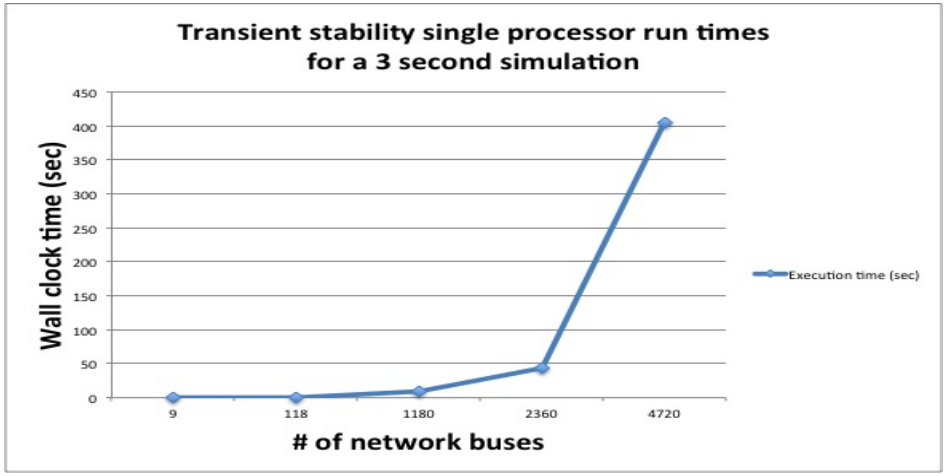
One possible solution: Parallel computing

Need to develop parallel scientific computational tool (solver) for specific application (physics) ..... not so easy!!

High performance computing libraries, such as PETSc, can aid in

- Reduce the computational time
- Rapid development of parallel applications
- Reduce the experimentation time and effort

Computational challenges



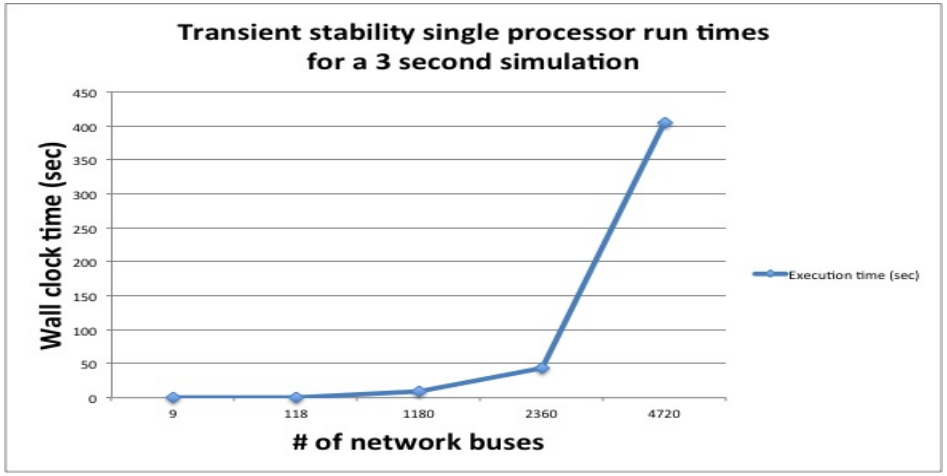
One possible solution: Parallel computing

Need to develop parallel scientific computational tool (solver) for specific application (physics) ..... not so easy!!

High performance computing libraries, such as PETSc, can aid in

- Reduce the computational time
- Rapid development of parallel applications
- Reduce the experimentation time and effort

Computational challenges



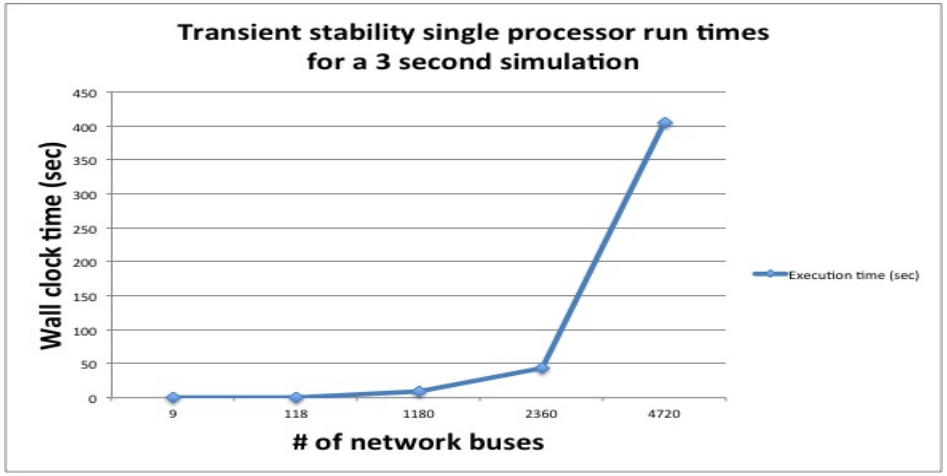
One possible solution: Parallel computing

Need to develop parallel scientific computational tool (solver) for specific application (physics) ..... not so easy!!

High performance computing libraries, such as PETSc, can aid in

- Reduce the computational time
- Rapid development of parallel applications
- Reduce the experimentation time and effort

Computational challenges



One possible solution: Parallel computing

Need to develop parallel scientific computational tool (solver) for specific application (physics) ..... not so easy!!

High performance computing libraries, such as PETSc, can aid in

- Reduce the computational time
- Rapid development of parallel applications
- Reduce the experimentation time and effort

Outline

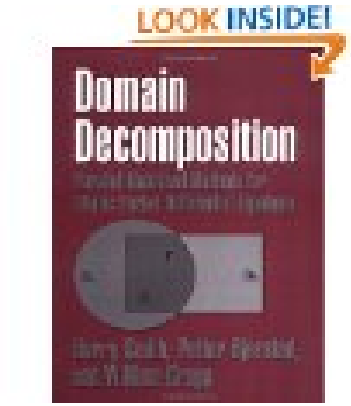
What is PETSc?

Portable Extensible Toolkit for Scientific Computation

- High performance library for the scalable (parallel) solution of scientific applications
- Developed at Argonne National Laboratory
- Mostly used by researchers in PDE applications
- Free for anyone to use, including industrial users
- Download from http://www.mcs.anl.gov/petsc
- Funding
  - Department of Energy
  - National Science Foundation

History of PETSc

- Begun in September 1991 as a platform for experimentation



- More than 60,000 downloads since 1995 (version 2)
- About 400 downloads per month
- Awards
  - Top 100 R & D award in 2009
  - Cited as the Top 10 computational science accomplishments of DOE in 2008

Portable Extensible Toolkit for Scientific computing

- Architecture
  - tightly coupled (e.g. Cray XT5, BG/P, Earth Simulator)
  - loosely coupled (network of workstations)
  - GPU clusters (many vector and sparse matrix kernels)
- Operating systems (Linux, Unix, Mac, Windows)
- Any compiler
- Real/complex, single/double/quad precision, 32/64-bit int
- Usable from C, C++, Fortran 77/90, Python, and MATLAB
- Free to everyone, open development, including industrial users

Portable **Extensible** Toolkit for Scientific computing

Interface for other HP libraries

- BLAS, LAPACK, BLACS, ScaLAPACK, PLAPACK
- MPICH, MPE, Open MPI
- ParMetis, Chaco, Jostle, Party, Scotch, Zoltan
- MUMPS, Spooles, SuperLU, SuperLU_Dist, UMFPack, pARMS
- PaStiX, BLOPEX, FFTW, SPRNG
- Prometheus, HYPRE, ML, SPAI
- Sundials
- HDF5, Boost

Packages can be directly downloaded and installed at configure time
`--download-<packagename>=1`

Who uses PETSc?

- Computational Scientists
  - PyLith (CIG), Underworld (Monash), Magma Dynamics (LDEO, Columbia), PFLOTRAN (DOE), SHARP/UNIC (DOE)
- Algorithm Developers (iterative methods and preconditioning)
- Package Developers
  - SLEPc, TAO, Deal.II, Libmesh, FEniCS, PETSc-FEM, MagPar, OOFEM, FreeCFD, OpenFVM
- Hardware and software vendors
  - Cray and SiCortex
  - Fluent, Tech-X, Actel

What can we handle?

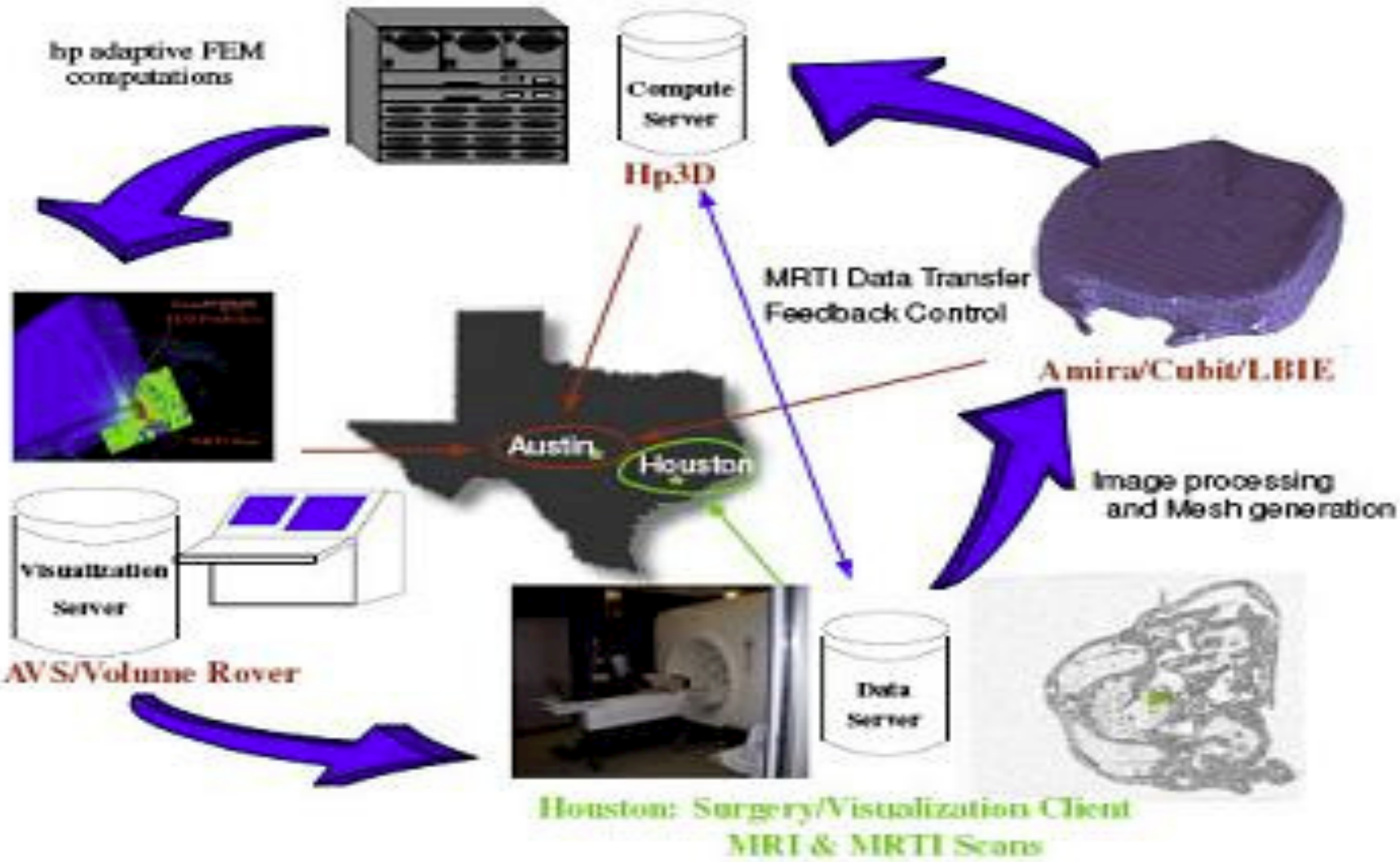
- PETSc has run implicit problems with 1 billion unknowns
  - PFLOTRAN for flow in porous media
- PETSc has run on over 224, 000 cores efficiently
  - UNIC on the IBM BG/P at ANL
  - PFLOTRAN on the Cray XT5 Jaguar at ORNL
- PETSc applications have run at 3 Teraflops
  - LANL PFLOTRAN code

Outline

Applications of PETSc

- Nano-simulations
- BiologyMedical
- Cardiology
- Imaging and Surgery
- Fusion
- Geosciences
- Environmental/Subsurface Flow
- Computational Fluid Dynamics
- Wave propagation and the Helmholz equation
- Optimization
- Fast Algorithms
- Software engineering
- Algorithm analysis and design
- Electrical Power Systems

Real-time laser surgery



Estimating uranium concentration

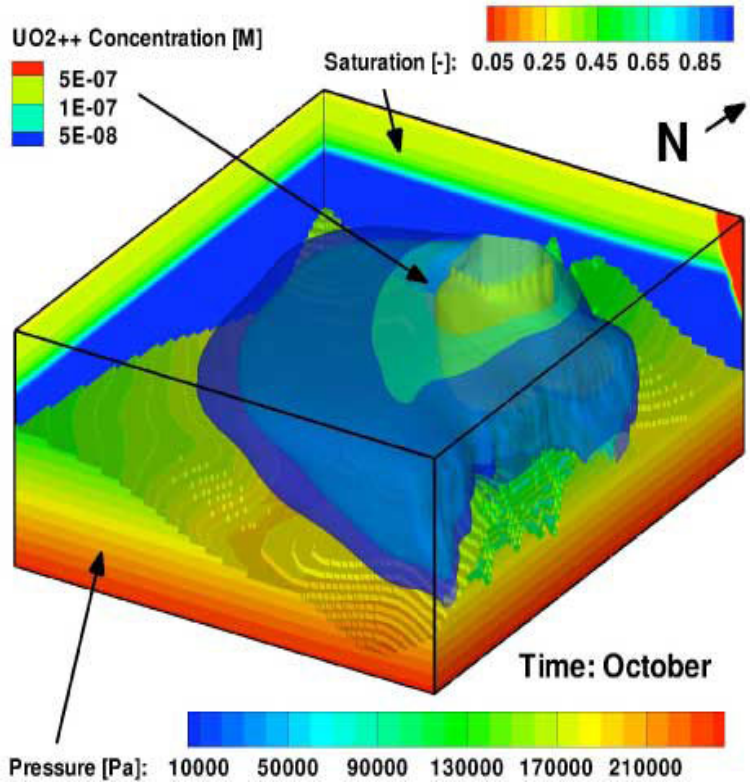


Figure: Isopleths of uranium calculated using PFLOTRAN at the Hanford 300 Area

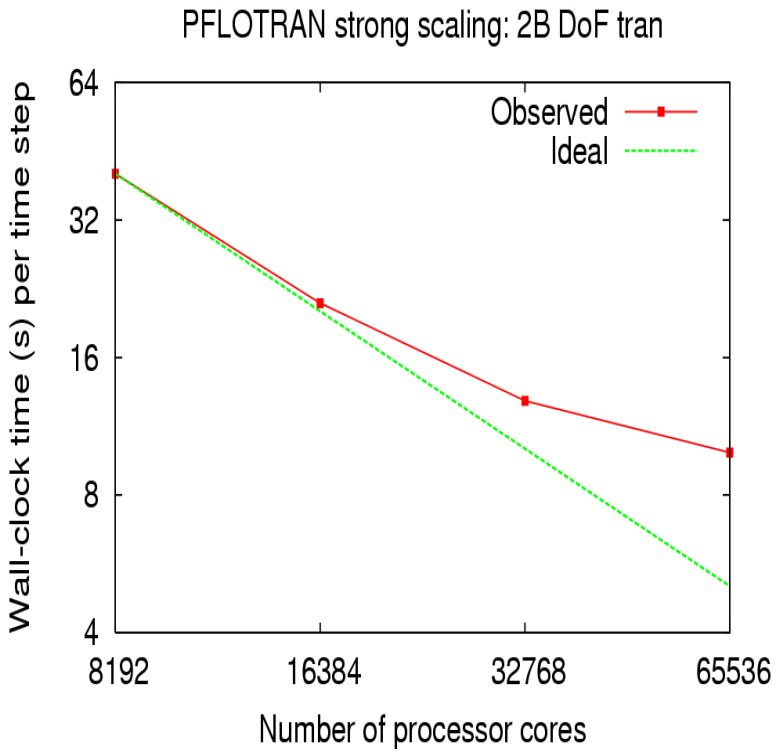


Figure: PFLOTRAN scaling for 2B degrees of freedom

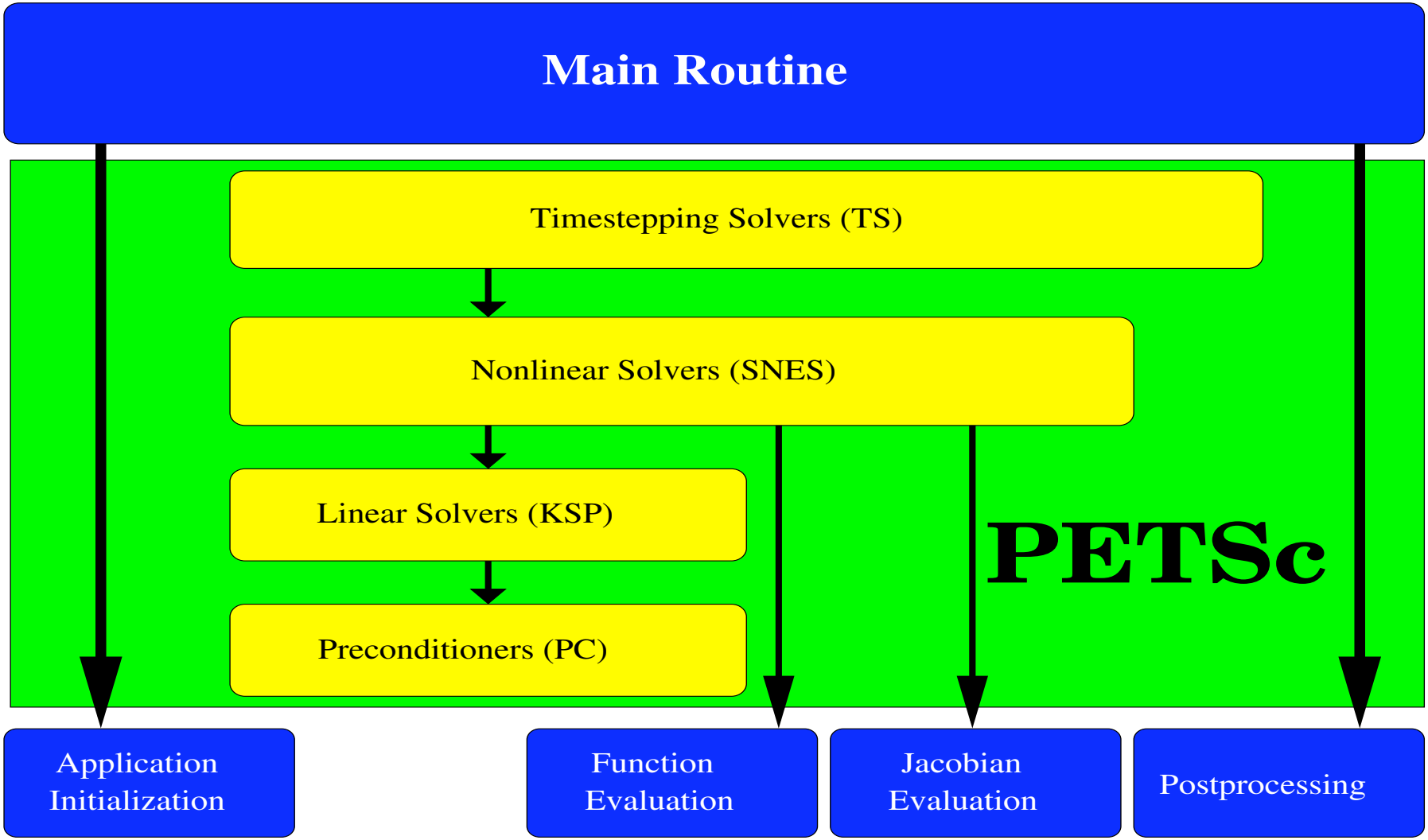
Outline

Design principles

- Linear algebra interface (Vectors, Matrices, Index sets)
- Distributed, shared nothing
  - User orchestrates communication through higher level interface
  - You almost never will have to use MPI directly
- Object-oriented design
  - Design based on the operations you perform
  - Example : A vector is
    - not a 1-d array but
    - an object allowing addition and scalar multiplication
- Polymorphism
  - User does not need to know the underlying implementation
- Allow solver composition to be set at run-time
  - Great for experimentation

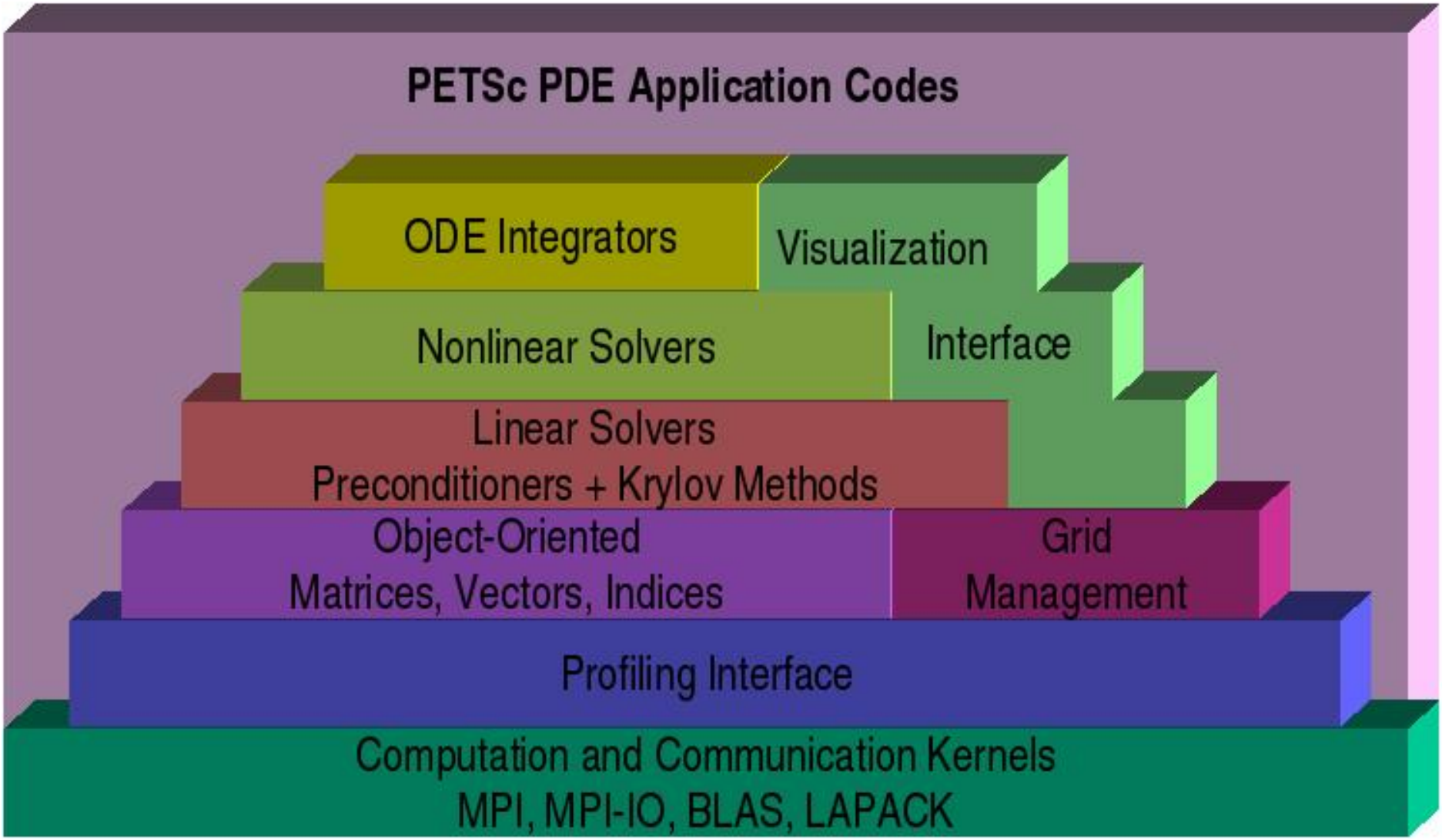
```
./ex -snes_type <ls,tr,test> -ksp_type
<gmres,cg,bicg,preonly> -pc_type <lu,ilu,icc,jacobi>
-mat_type <aij,baij,sbaij>
```

Flow control of a PETSc application



Outline

Library Organization



Numerical Components

Parallel Numerical Components of PETSc

Nonlinear Solvers		
Newton-based Methods		Other
Line Search	Trust Region	

Time Steppers			
Euler	Backward Euler	Pseudo-Time Stepping	Other

Krylov Subspace Methods							
GMRES	CG	CGS	Bi-CG-Stab	TFQMR	Richardson	Chebychev	Other

Preconditioners						
Additive Schwarz	Block Jacobi	Jacobi	ILU	ICC	LU (sequential only)	Other

Matrices				
Compressed Sparse Row (AIJ)	Block Compressed Sparse Row (BAIJ)	Block Diagonal (BDiag)	Dense	Other

Vectors

Index Sets			
Indices	Block Indices	Stride	Other

Outline

Debugging

- Automatic generation of tracebacks
- Detection of memory corruption and leaks
- Optional user-defined error handlers
- Launch the debugger
  - `-start_in_debugger [gdb, dbx, noxterm]`
  - `-on_error_attach_debugger [gdb, dbx, noxterm]`
- Attach the debugger only to some parallel processes
  - `-debugger_nodes 0, 1`
- Use valgrind
  - `http://www.valgrind.org`
  - Checks memory access, cache performance, memory usage, etc.
- Check correctness of analytical Jacobian
  - `-snes_type test -snes_test_display`

Profiling

- `-log_summary`
  - Prints a report at the end of the run
  - Reports time, calls, Flops for function calls (called Events)
  - Memory usage for Objects
  - Can set stages for code profiling

Sample -log_summary

```
------------------------------------------------------------------------------------------------------------------------
Event                Count      Time (sec)     Flops/sec                         --- Global ---  --- Stage ---   Total
                   Max Ratio  Max     Ratio   Max  Ratio  Mess   Avg len Reduct  %T %F %M %L %R  %T %F %M %L %R Mflop/s
------------------------------------------------------------------------------------------------------------------------

--- Event Stage 0: Main Stage

PetscBarrier           2 1.0 1.1733e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0

--- Event Stage 1: SetUp

VecSet                 2 1.0 9.3448e-04 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0
MatMultTranspose       1 1.0 1.8022e-03 1.0 1.85e+08 1.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0 57  0  0  0   185
MatAssemblyBegin       3 1.0 1.0057e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0
MatAssemblyEnd         3 1.0 2.0356e-02 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   5  0  0  0  0     0
MatFDColorCreate       2 1.0 1.5341e-01 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 4.6e+01  1  0  0  0 16  36  0  0  0 74     0

--- Event Stage 2: Solve

VecDot                 2 1.0 3.2985e-03 1.0 9.56e+07 1.0 0.0e+00 0.0e+00 2.0e+00  0  0  0  0  1   0  0  0  0  2    96
VecMDot               45 1.0 9.3093e-02 1.0 1.59e+08 1.0 0.0e+00 0.0e+00 1.5e+01  0  0  0  0  5   1  1  0  0 19   159
VecNorm              112 1.0 2.0851e-01 1.0 8.47e+07 1.0 0.0e+00 0.0e+00 5.2e+01  1  1  0  0 18   2  1  0  0 64    85
```

Outline

Splitting for Multiphysics

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

- Relaxation: `-pc_fieldsplit_type [additive,multiplicative,symmetric_multiplicative]`

$$\begin{bmatrix} A & \\ & D \end{bmatrix}^{-1} \qquad \begin{bmatrix} A & \\ C & D \end{bmatrix}^{-1} \qquad \begin{bmatrix} A & \\ & \mathbf{1} \end{bmatrix}^{-1} \left(\mathbf{1} - \begin{bmatrix} A & B \\ & \mathbf{1} \end{bmatrix} \begin{bmatrix} A & \\ C & D \end{bmatrix}^{-1} \right)$$

  - Gauss-Seidel inspired, works when fields are loosely coupled
- Factorization: `-pc_fieldsplit_type schur`

$$\begin{bmatrix} A & B \\ & S \end{bmatrix}^{-1} \begin{bmatrix} 1 & \\ CA^{-1} & 1 \end{bmatrix}^{-1}, \qquad S = D - CA^{-1}B$$

Python bindings and MATLAB interface

- Python bindings (petsc4py)
  - Implemented with Cython
  - Easier to write code, maintain, and extend
  - Supports all PETSc libraries
  - `http://code.google.com/p/petsc4py`
- PETSc-MATLAB interface
  - PETSc functions can be called via MATLAB code.
  - Supports almost all PETSc functionalities
  - Uses 1-based indexing (consistent with MATLAB)

Memory-efficient LU factorization

- Revise LU data structure according to the elements accessed during triangular solves
- Store L forward followed by U backwards
- Provides better Cache performance

`Typical LU data structure`

$$[L(1,:), U(1,:), L(2,:), U(2,:), ..., L(n,:), U(n,:)]$$

`Revised LU data structure`

$$[L(1,:), L(2,:), ..., L(n,:), U(n,:), ..., U(2,:), U(1,:)]$$

Support for new architectures

- Graphical Processing Units (GPU)
  - PETSc-3.2 (current version) supports computations on the NViDia GPUs
  - Uses CUSP and Thrust libraries provided by NViDia guys
  - Vec and Mat classes implemented on the GPU
  - Krylov solvers come for free
- Hybrid MPI-shared memory architectures
  - Shared-memory implementation of Vec and Mat using POSIX pthreads

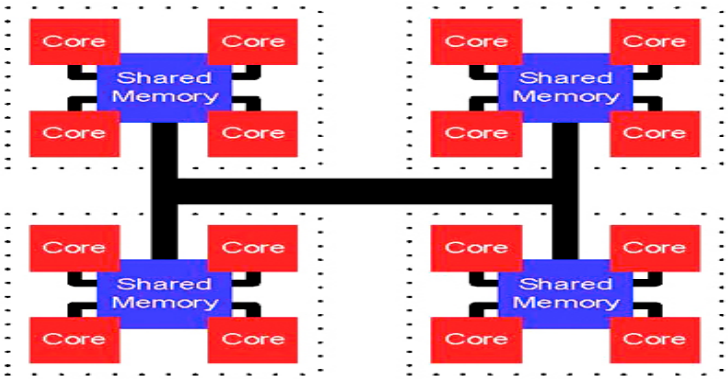


Figure: Hybrid MPI-Shared memory

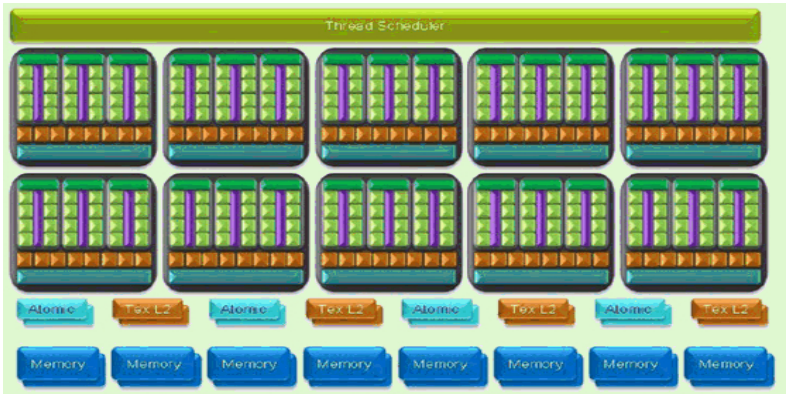


Figure: NViDia GTX 280 GPU

Variational Inequalities

$$f(x) = 0$$
$$s.t. \qquad xl <= x <= xu$$

- Supports inequality and box constraints on solution variables.
- Solution methods
  - Semismooth Newton
    - reformulate problem as a non-smooth system, Newton on sub-differential
    - Newton step solves diagonally perturbed systems
  - Active set
    - solve in reduced space by eliminating constrained variables
    - or enforce constraints by Lagrange multipliers
    - sometimes slower convergence or "bouncing"

Outline

Power system applications

- Linear (KSP library)
  - DC Power Flow, Sensitivity factors
- Nonlinear (SNES library)
  - AC Power Flow, Contingency analysis, Continuation power flow
  - Distribution power flow, Combined Transmission-distribution power flow
- Time-stepping (TS library)
  - Transient stability, Electromagnetic transients
  - Combined transient stabilty-electromagnetic transients (hybrid simulation)
- Optimization (using TAO)
  - SCOPF
- Eigen-value analysis (using SLEPc)
  - Small signal stability analysis

Outline

Transient Stability Simulators (TS)

- For studying relatively slow dynamic behavior (generator dynamics)
- Assumptions
  - Constant frequency (phasor representation of voltages/currents)
    - Can use larger time step (in the order of milliseconds)
    - Allow analysis of large-scale systems.
  - Balanced transmission network (positive sequence network)
- Limitations of balanced transmission network
  - Unbalanced operation or systems
  - Single phase switching operations
  - Only positive sequence information available

Three phase dynamics simulator (TS3ph)

- Full three-phase coupled network model of the transmission network.
- Still uses constant frequency assumption as in TS.
- Can additionally
  - Simulate unbalanced operation
  - Single phase switching operations
  - Provide information on dynamics of all three phases
  - Model individual load dynamics on different phases
  - Can be also used for distribution systems

Problem formulation

- Nonlinear differential-algebraic three-phase power system model

$$\frac{dx_{gen}}{dt} = f(x_{gen}, I_{dq}, V_{DQ,abc})$$

$$0 = h(x_{gen}, I_{dq}, V_{DQ,abc})$$

$$\begin{bmatrix} G_{3ph} & -B_{3ph} \\ B_{3ph} & G_{3ph} \end{bmatrix} \begin{bmatrix} V_{D,abc} \\ V_{Q,abc} \end{bmatrix} = \begin{bmatrix} I_{genD,abc}(x_{gen}, I_{dq}) \\ I_{genQ,abc}(x_{gen}, I_{dq}) \end{bmatrix} - \begin{bmatrix} I_{loadD,abc}(x_{load}, V_{DQ,abc}) \\ I_{loadQ,abc}(x_{load}, V_{DQ,abc}) \end{bmatrix}$$

$$\frac{dx_{load}}{dt} = f_2(x_{load}, V_{DQ,abc})$$

- Equations to solve at each time step

$$x(t+\Delta t) - x(t) - \frac{\Delta t}{2}\left(f(x(t+\Delta t), y(t+\Delta t)) + f(x(t), y(t))\right) = 0$$

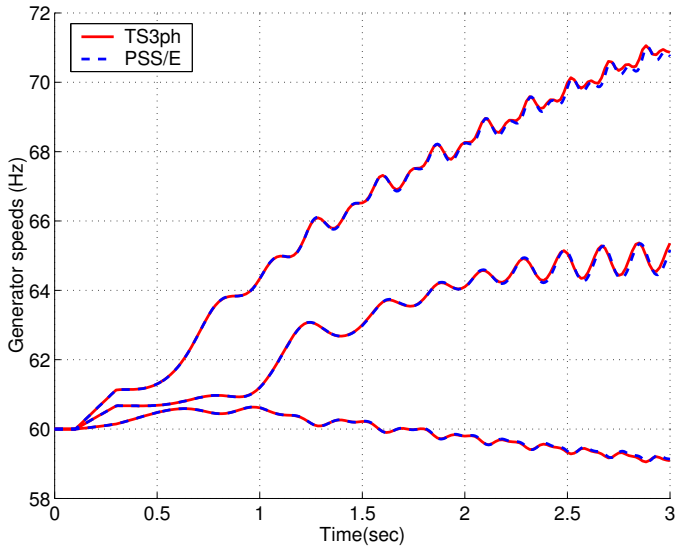
$$g(x(t+\Delta t), y(t+\Delta t)) = 0$$

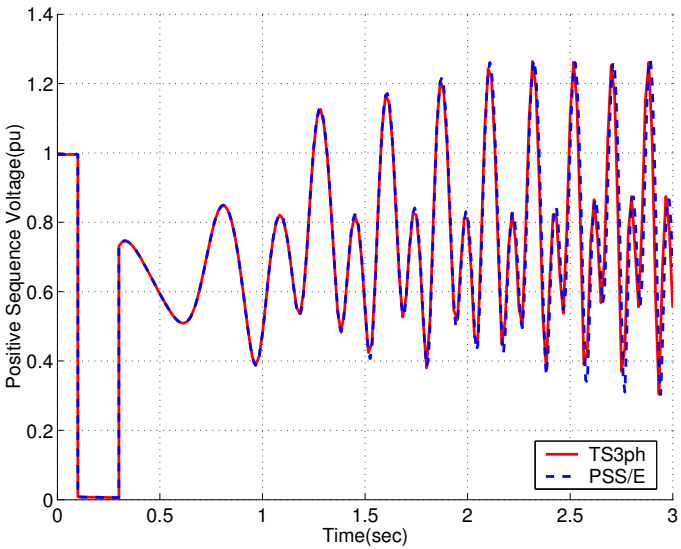
- Variables

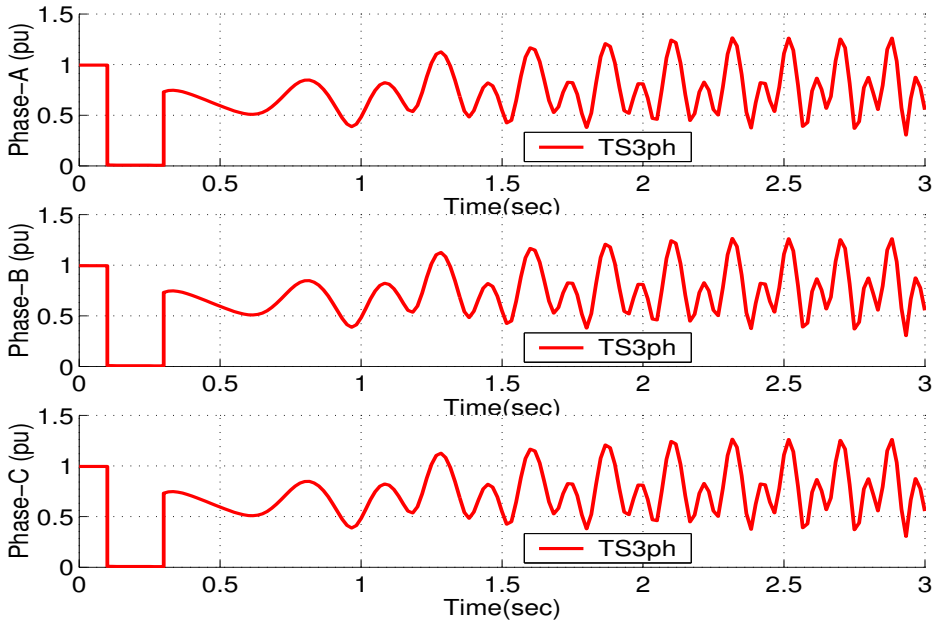
$$x \equiv [x_{gen}, x_{load}]^t$$

$$y \equiv [I_d, I_q, V_{D,abc}, V_{Q,abc}]^t$$

TS3ph benchmarking with PSS/E on WECC 9-bus system

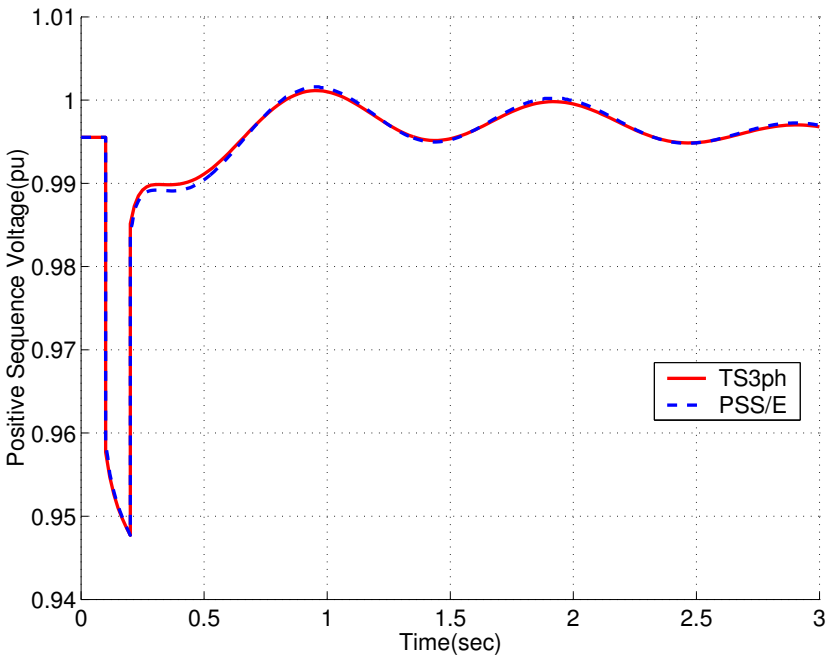




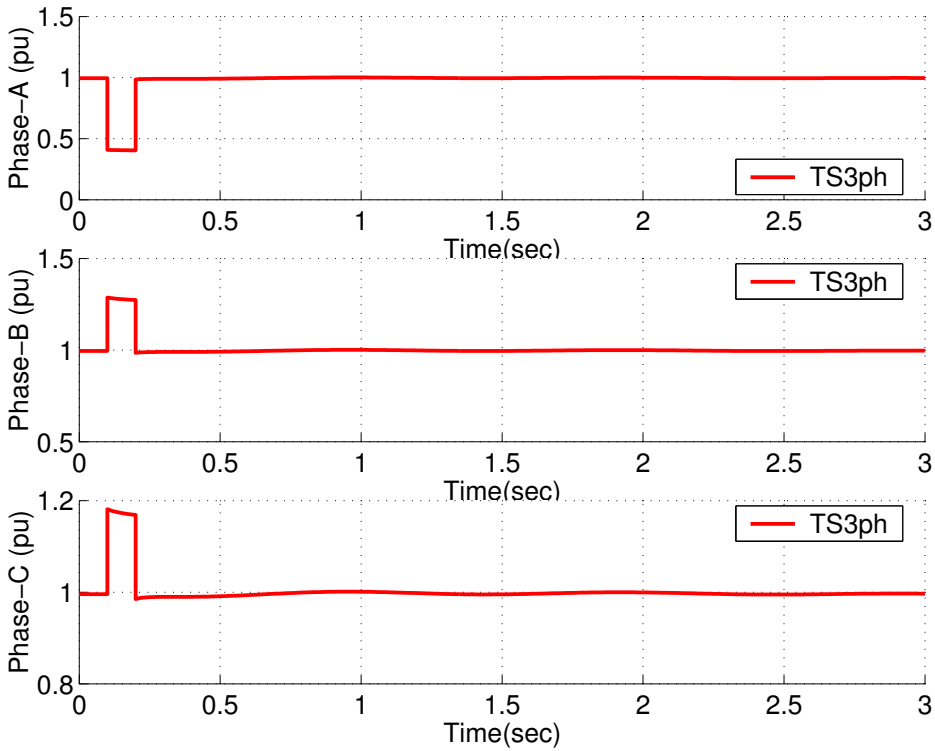


Unbalanced faults

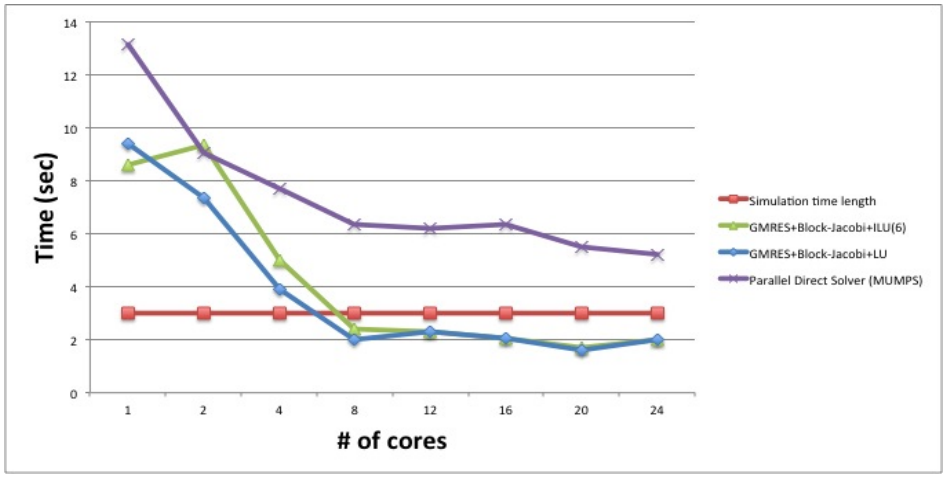
Positive-sequence voltages

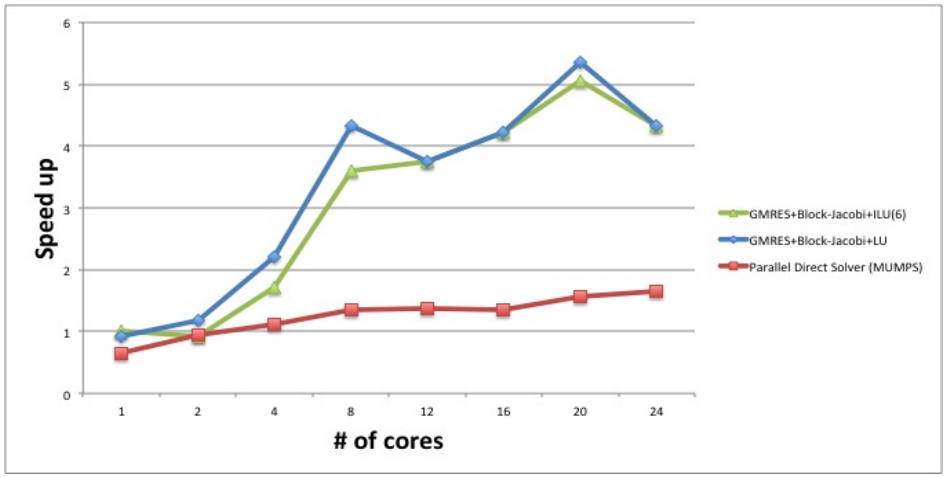


Three-phase voltages



Parallel performance of TS3ph

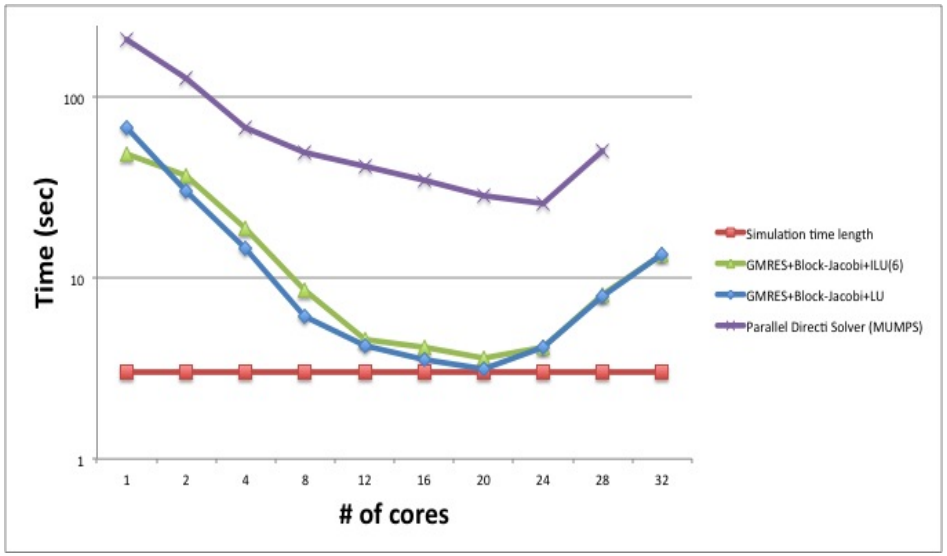


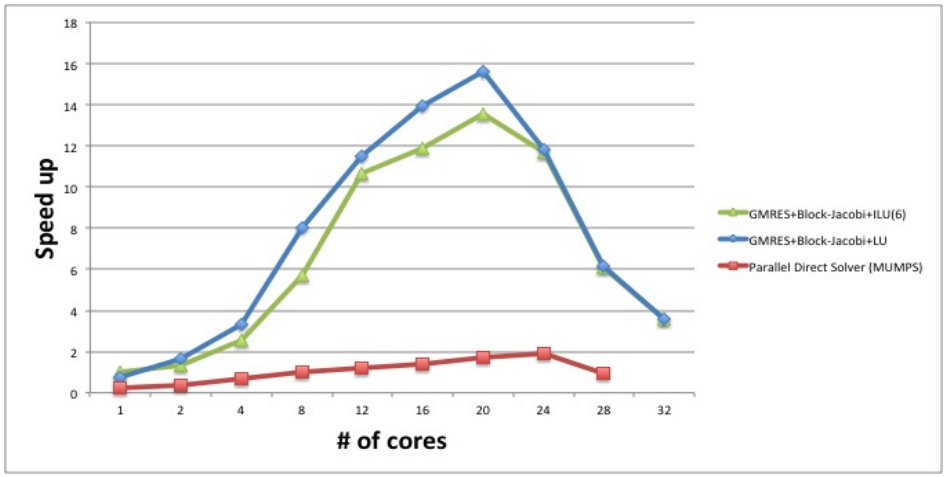


*Scalabilty results of 1180 bus, 2035 branches, 540 generator system

- Parallel linear solves too expensive!!
- Results of experiments to speed up linear solves
  - Preconditioner Iterative solver GMRES more scalable than parallel direct solver (MUMPS)
  - Parallel Block-Jacobi (Block-Diagonal) preconditioning
  - Very dishonest preconditioning strategy (Pai et. al.)

Parallel performance of TS3ph





- Scalabilty results of 2360 bus, 4670 branches, 1080 generator system

Power system dynamic analysis methods

- Electromechanical dynamics simulation
  - Assess "slow" generator dynamics
  - Time step in the order of milliseconds
  - Phasor modeling
  - Can be used for large-scale system analysis.
- Electromagnetic transients simulation
  - Analyze "faster" dynamics such as that of power electronics
  - Time step in the order of microseconds
  - No constant frequency assumption
  - Inefficient for large-scale system analysis
- Combined electromechanical-electromagnetic transients simulator ("hybrid" simulator)
  - Capture "global" slow dynamics and "local" fast dynamics
  - Use TS globally and EMT locally
  - Need interface for
    - Time step
    - Network modeling
    - Waveform

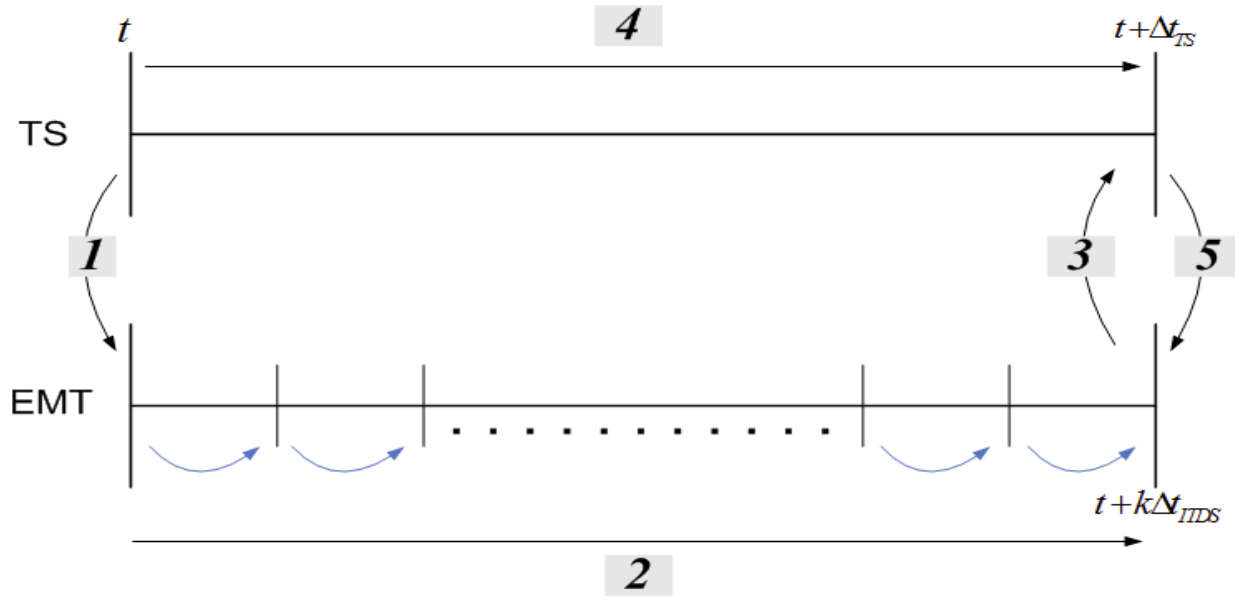
Power system dynamic analysis methods

- Electromechanical dynamics simulation
  - Assess "slow" generator dynamics
  - Time step in the order of milliseconds
  - Phasor modeling
  - Can be used for large-scale system analysis.
- Electromagnetic transients simulation
  - Analyze "faster" dynamics such as that of power electronics
  - Time step in the order of microseconds
  - No constant frequency assumption
  - Inefficient for large-scale system analysis
- Combined electromechanical-electromagnetic transients simulator ("hybrid" simulator)
  - Capture "global" slow dynamics and "local" fast dynamics
  - Use TS globally and EMT locally
  - Need interface for
    - Time step
    - Network modeling
    - Waveform

Power system dynamic analysis methods

- Electromechanical dynamics simulation
  - Assess "slow" generator dynamics
  - Time step in the order of milliseconds
  - Phasor modeling
  - Can be used for large-scale system analysis.
- Electromagnetic transients simulation
  - Analyze "faster" dynamics such as that of power electronics
  - Time step in the order of microseconds
  - No constant frequency assumption
  - Inefficient for large-scale system analysis
- Combined electromechanical-electromagnetic transients simulator ("hybrid" simulator)
  - Capture "global" slow dynamics and "local" fast dynamics
  - Use TS globally and EMT locally
  - Need interface for
    - Time step
    - Network modeling
    - Waveform

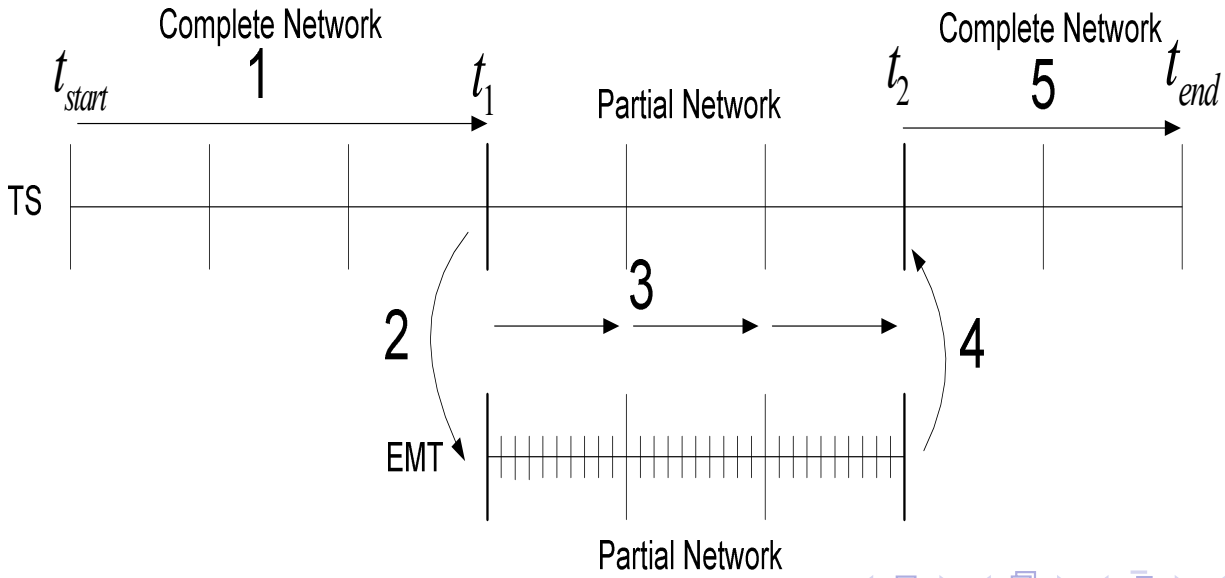
Existing "explicit" hybrid simulation approach



- Make separate TS and EMT programs talk to each other
- Explicit approach
- No iterations between TS and EMT
- Diverges for large changes in voltages/currents

Proposed "implicitly-coupled" hybrid simulation approach

- Combine TS and EMT at the equation level rather than at the application level
- Solve TS equations and coupled-in-time EMT equations for each TS time step together
- More robust than the explicit approach
- Allows a parallel implementation
- Proposed multi-scale dynamics simulation strategy
  - Only run hybrid simulator when needed, run TS for all other times



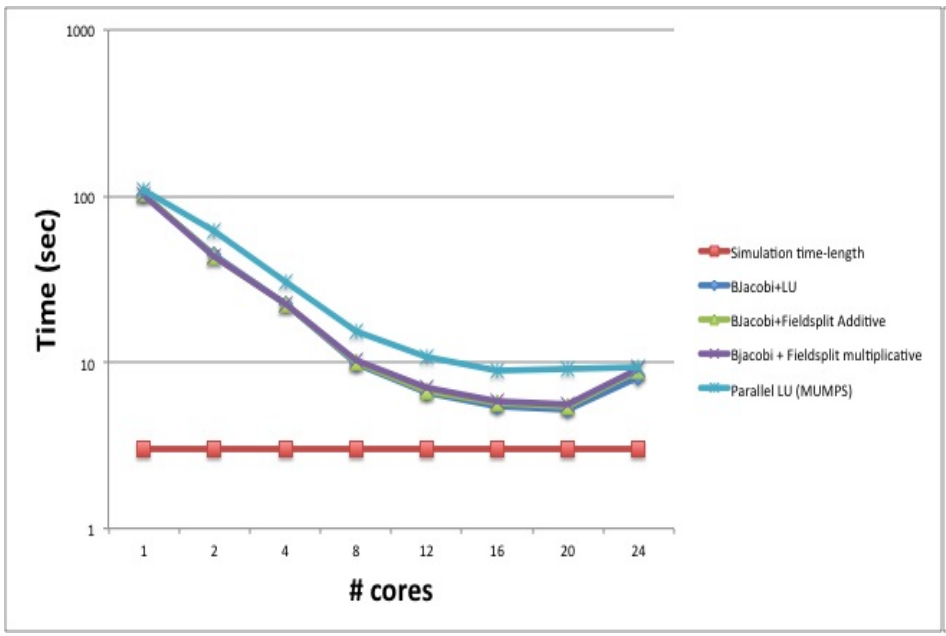
Combined electromechanical-electromagnetic transients simulation

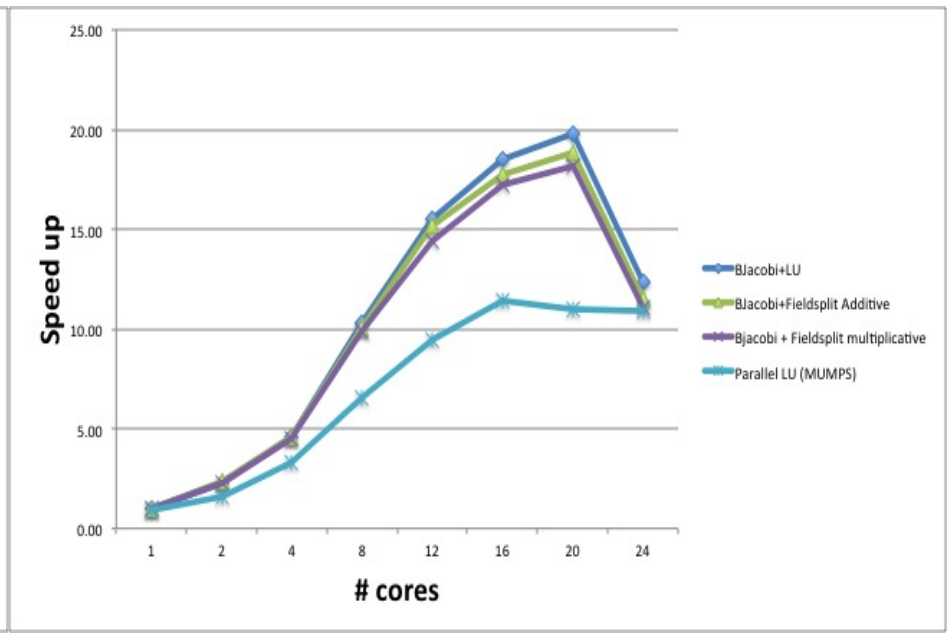
- Time-comparison of different dynamic analyses

System size	Simulated time (sec)	TS3ph	EMT	Only TSEMT	TS3ph-TSEMT
9 bus	3	0.13	4.96	5.46	0.41
118 bus	3	0.36	30.1	4.87	0.53

Parallel implementation and performance results

- Partition TS network in space and EMT network in time
- Each processor gets equations for
  - TS subnetwork
  - EMT equations for multiple time-steps





*2360 buses total, 4 buses,3 transmission lines and 4 loads in EMT network

*Using GMRES + Very Dishonest preconditioning

PETSc use in the example applications

- Easy parallel implementation
- Partitioning (using ParMetis)
- Linear solver (using KSP and PC libraries)
- Nonlinear solver (SNES library)
- Portable code
- Reduced experimentation time
  - Set different algorithms at run-time!!

The Role of PETSc

Developing parallel, nontrivial applications that deliver high performance is still difficult and requires months (or even years) of concentrated effort.

PETSc is a toolkit that can ease these difficulties and reduce the development time, but it is not a black-box solver, nor a silver bullet.

— Barry Smith